# InterActive 1

Written By: Steven Robert Cypherd

## SUMMARY

Life needs interaction to live. Machines need programs to interact with us. Programs need connectivity to grow more complicated. Serial communication is what machines use to grow. Serial communication is built into most processors. That is how you first program them and debug them. Serial connections let each part of a machine talk to the other parts to make the machine do more complicated tasks. People love interactive things. Serial connections let you make your machine interactive. I will show you an example of an interactive machine using my EasyLights and buttons.

[<iframe width="420" height="315" src="http://www.youtube.com/embed/lUXusYytw00... frameborder="0" allowfullscreen></iframe>]

Getting my Basic Stamps to talk took a little programming. I wanted to run EasyLights in my Basic Stamp 2, sending the lights byte to the Basic Stamp 1 and reading buttons back. I had trouble getting messages between the Basic Stamp 2 (BS2) and the Basic Stamp 1 (BS1). The BS2 is about four times faster than the BS1. The BS1 locks up waiting for the BS2 message. The BS1 cannot see the BS2 message in the 200 millisecond (ms) window it has to check for a message. This is exactly why there are so many flow control lines left in standard serial connectors. Slower machines can't see faster ones. I needed flow control, but I didn't want to use another port for it. I have an electronics background so I looked at the Rx and Tx as ports, not just as serial connections.

Steve's Flow Control uses the Tx port as flow control. You hold your Tx port high with a resistor of 47K to VDD and in your code. You monitor your Rx port to detect when someone lowers its Tx port. When you want to send a message you bring your Tx port low, wait until

the receiver lowers its Tx port and then you send your message. The receiver should stay in a tight loop waiting for the message. After the message both systems raise their Tx ports high and go back to waiting.

For the BS1 and slower systems I lower my (Master) BS2's Tx port and pause about 80 ms, and then I transmit my message. The BS1 looks for a low on its Rx port and pauses 1ms and then opens SerIn for a two-byte message. The first byte is the Address and the second byte is the Lights. If the Address is not correct the BS1 skips the message and goes back to waiting. The BS1 is too slow for full handshaking. It works perfectly with correct timing.

I can only send one message to the BS1 in every ten loops of my BS2's code. I am running EasyLights in my BS2 and sending the lights byte to the BS1 that only has doLights and checkLights and a message loop. See my EasyLights article.

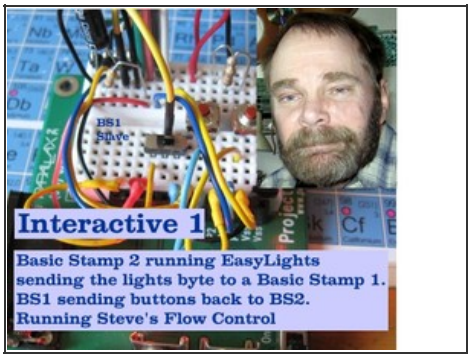
I could not get the BS1 to send a correct two-byte message. The first byte is garbled and I do not know why. The BS1 can't use my standard two-byte messaging (Address, Data) so I am using a separate serial channel to handle this type of slave. The BS1 lowers its Tx port and waits until the Master lowers its slave Tx port and then the BS1 sends a one-byte message containing the button flags to the Master. They both raise their Tx ports high and go back to waiting.

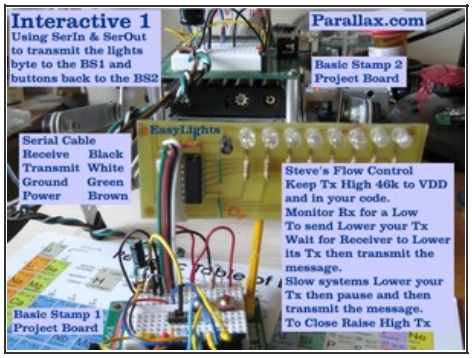
Working with the BS1 I decided to make them polled slaves. I simply keep them alive by constantly sending them messages. They do all of their work inside their message loop. There is no room for much else. They just sit there and wait for a message so their buttons and lights are, in effect, dead. Polling them keeps their buttons and lights alive.

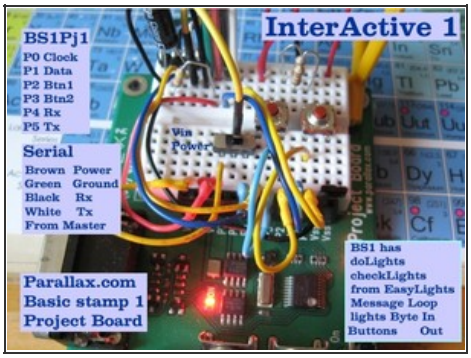
As I get things done I will upload another part to this interactive article.

See articles and source code on http://letsmakerobots.com and http://instructables.com.

## Step 1 — InterActive 1

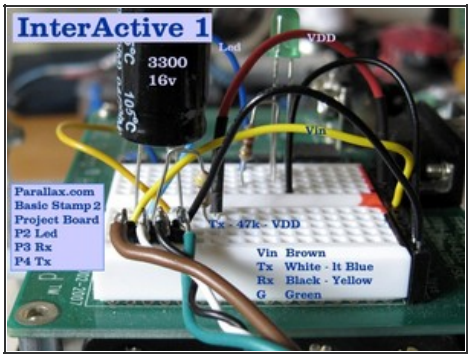



- No circuit.

## Step 2




- The Basic Stamp 2
- More documents

## Step 3

- Last one

Fun way to get machines talking.

This document was last generated on 2012-11-02 04:42:17 AM.